REMARKS

This paper is responsive to a non-final Office action dated April 21, 2004. Claims 1-21 and 29-35 were examined. Claims 22-28 have been withdrawn. Applicant respectfully traverses all rejections.

Election of Invention

In response to the Examiner's restriction requirement, the undersigned hereby affirms election of Group I, pending claims1-21 and 29-35.

Rejections under 35 U.S.C. §102(b)

The Office Action rejects claims 1, 2, 7 – 9, 13 – 17, 20, 29, 30 and 33 - 35 under 35 U.S.C. §102(b) as being anticipated by "Managing Long Linked Lists Using Lock-Free Techniques" by Mohammad Farook and Peter Graham ("Farook"). Applicant respectfully traverses all rejections.

One of the flaws of Farook is that Farook's reference count only reflects the number of processes currently accessing a shared node of a list. Farook's reference counter does not reflect the state or structure of the list. Farook suffers from flaws that allows for premature reclamation of a shared object and additional failures. Premature reclamation of an object can result in arbitrary failure. Although Farook states that a "process attempting a node deletion must first verify that its counter field is zero before proceeding", Farook's technique allows for a discrepancy between an object's reference counter and the actual number of referencing pointers. **As the Office Action notes in rejecting claim 3, concurrent processes may simultaneously read an object's reference counter and simultaneously increment the reference counter, resulting in the object's reference counter being two instead of three (three being the correct reference counter value).** Allowing the actual number of referencing pointers to exceed the reference counter provides an opportunity for premature reclamation. The accessing processes may then decrement the reference counter from two to zero. **Although a pointer still references the object, the reference counter indicates that the referenced object is unreferenced.** In addition to premature reclamation, the operations disclosed by Farook may decrement the reference counter to a negative value, thus propagating failures (e.g., the object cannot be deleted when attempted because TryDelete will decrement the reference counter to a negative value). Moreover, Farook's TryDelete allows release of a node that is being accessed. A first process checks the counter of a target node prior to freeing the target node. However, there is nothing in Farook that prevents a second process from accessing the target node after the first process verifies that the counter is zero. **Hence, Farook allows a first process to free a shared node that is being accessed by a second process.**

The described invention does not allow for premature reclamation. Farook's reference counter fails to reliably indicate whether a shared object is unreferenced and especially does not disclose or suggest "freeing storage associated with a particular one of the shared objects only once the corresponding reference count indicates that the particular shared object is unreferenced" as recited in claim 1, and similarly recited in independent claims 29 and dependent claim 35. In addition, Farook does not disclose or suggest "means for coordinating competing access to the shared object using one or more reference counts and pointer manipulations that employ one or more lock-free pointer operations to ensure that if the number of pointers to the shared object is non-zero, then so too is the corresponding reference count and further that if no pointers reference the shared object, then the corresponding reference count eventually becomes zero" as recited in claim 34, and similarly recited in dependent claim 2. As discussed above, Farook allows for a shared object's reference count to be zero even if there are pointers referencing the shared object, and allows the reference count to be non-zero even if there are no pointers referencing the shared object.

The unreliability of Farook's reference counters at least arises from Farook's timing of modifying the reference counters in relation to creating and destroying pointers to shared objects. Claim 13 recites the following:

access operations that, prior to attempting creation or replication of a pointer to any of the component shared objects, increment a corresponding reference count, and upon failure of the attempt, thereafter decrement the corresponding reference count, the access operations decrementing a particular reference count…no earlier than upon destruction of a pointer to a corresponding one of the component shared objects.

Neither temporal relationship recited in claim 13 is disclosed or suggested in Farook. Farook does not disclose or suggest incrementing a reference count prior to creation (or replication) of a pointer to a shared object, and does not disclose or suggest decrementing a reference count no earlier than upon destruction of a pointer to a corresponding shared object. Figure 9 of Farook illustrates code for TryInsert that initializes the reference count to zero when attempting to insert a node into a list and does not increment the reference count. If the attempt to insert the node fails, then Farook attempts the insert again. Upon successful insertion of the node, Farook decrements a reference counter (the decremented reference counter being that for the node that precedes the inserted node).

For at least the reasons discussed above, Applicant's independent claims 1, 13, 29 and 34, and dependent claims 2 and 35 are allowable and not anticipated by Farook. In addition, claims 3 – 12, 14 – 21, and 30 – 33 are dependent are corresponding ones of the above allowable independent claims.

Dependent claims

With regard to claim 7, Farook does not disclose or suggest "a destroy operation that decrements a reference count of a shared object identified by a supplied pointer value; and frees the identified shared object if the corresponding reference count has reached zero being inserted to zero" as recited in claim 7. The TryDelete operation, referred to by the Office Action, decrements a counter of a node that precedes the node being deleted. None of the operations in Farook actually decrement the counter of the target node.

With regard to claim 8, the TryDelete operation in Farook relied upon by the Office Action does not recursively follow pointers of a target node that the TryDelete operation is attempting to delete. The TryDelete operation checks the pointers of the target node to determine if the pointers have changed and returns a TRY_AGAIN flag if those pointers have changed.

With regard to claim 15, Farook does not disclose or suggest the operations recited in claim 15. The Office Action states suggests that the TryDelete operation is the same as "a lock-free reference counting maintaining destroy operation" as recited in claim 15. Even if Farook's TryDelete operation is the same as the recited destroy operation, which it is not, Farook does not disclose or suggest the other operations recited in claim 15.

With regard to claim 20, Farook does not disclose incrementing and decrementing reference counters using a synchronization primitive. Farook conditions decrementing a reference counter on the return value of a DCAS in TryDelete, but does not perform the decrementing with a synchronization primitive. For incrementing reference counters, Farook does not event condition the incrementing on a synchronization primitive, much less perform the incrementing with a synchronization primitive.

Rejections under 35 U.S.C. §103(a)

The Office Action rejects claims 3 and 21 under 35 U.S.C. §103(a) as being unpatentable over Farook. The Office Action rejects claim 10 under 35 U.S.C. §103(a) as being unpatentable over Farook in view of "Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms" by Maged M. Michael and Michael L. Scott ("Michael"). The Office Action rejects claims 18 and 19 under 35 U.S.C. §103(a) as being unpatentable over Farook in view "Transactional Memory: Architectural Support for Lock-Free Data Structures" by Maurice Herlihy et al. ("Herlihy"). The Office Action rejects claims 31 and 32 under 35 U.S.C. §103(a) as being unpatentable over Farook in view of "Garbage Collection: Algorithms for Automatic Dynamic Memory Management" by Richard Jones and Rafael Lins ("Jones"). Applicant respectfully traverses all rejections.

With regard to claim 3, Farook does not disclose or suggest claim 3 as already discussed above.

With regard to claim 19, the Office Action does not provide a reference or assumption that discloses or suggests the emulation being based on "a load-linked/store-conditional operation pair" as recited in claim 19.

Farook, Michael, Herlihy, and Jones, standing alone or in combination, do not disclose or suggest any of Applicant's claims. Claims 3, 10, 18, 19, 21, 31, and 32 depend on one of the above allowable independent claims and are allowable at least for the reasons already discussed.

Conclusion

In summary, claims 1-21 and 29-35 are in the case. All claims are believed to be allowable over the art of record, and a Notice of Allowance to that effect is respectfully solicited. Nonetheless, if any issues remain that could be more efficiently handled by telephone, the Examiner is requested to call the undersigned at the number listed below.

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that, on the date shown below, this correspondence is being

☐ deposited with the US Postal Service with sufficient postage as first class mail, in an envelope addressed to Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

☒ facsimile transmitted to the US Patent and Trademark Office.

[illegible] Jul-21-2004
Steven R. Gilliam Date

EXPRESS MAIL LABEL: ____________

Respectfully submitted,

[illegible]

Steven R. Gilliam, Reg. No. 51,734
Attorney for Applicant(s)
(512) 338-6320
(512) 338-6301 (fax)